(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
20 February 2003 (20.02.2003)

PCT

(10) International Publication Number
WO 03/015292 A2

(51) International Patent Classification⁷: H04B

(21) International Application Number: PCT/US02/25138

(22) International Filing Date: 9 August 2002 (09.08.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/311,081 10 August 2001 (10.08.2001) US

(63) Related by continuation (CON) or continuation-in-part (CIP) to earlier application:
US 60/311,081 (CON)
Filed on 10 August 2001 (10.08.2001)

(71) Applicant *(for all designated States except US)*: ADAPTIVE NETWORKS, INC. [US/US]; 94 Wells Avenue, Newton, MA 02459 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: PROPP, Michael, B. [US/US]; 94 Wells Avenue, Newton, MA 02459 (US). SAAB, Khaled [CA/US]; 4 Liberty Place, Randolph, MA 02368-3726 (US).

(74) Agent: RENNER, W., Karl; Fish & Richardson P.C., 1425 K Street, NW, 11th Floor, Washington, DC 20005-3500 (US).

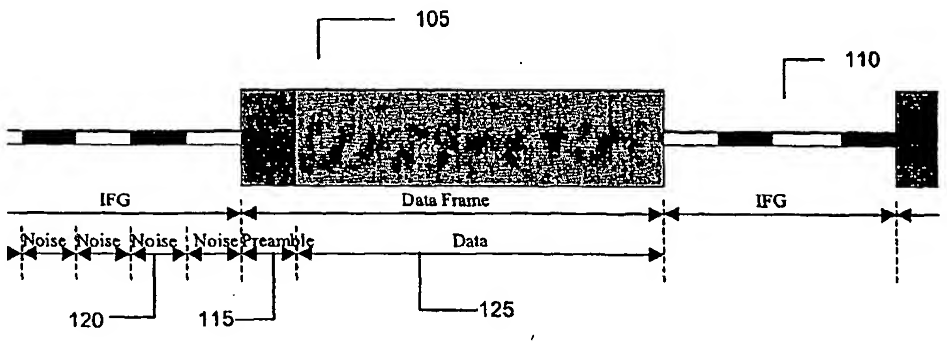
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: DIGITAL EQUALIZATION PROCESS AND MECHANISM



(57) Abstract: A filter is created by sampling noise during an inter-frame gap of a received signal, sampling a data frame preamble from within a data frame of the received signal, and computing filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

DIGITAL EQUALIZATION PROCESS AND MECHANISM

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims priority from U.S. Provisional Application No. 60/311,081, filed August 10, 2001, and titled "Digital Equalization Process and Mechanism," which is incorporated by reference.

TECHNICAL FIELD

This disclosure relates to data communications over noisy media with frequency-dependent attenuation.

BACKGROUND

In data communications, wideband transmission may be used. However, the received signal may be impaired by noise and frequency-dependent channel attenuation. For example, an entire portion of the transmitted signal may fall into an attenuation null and be severely attenuated. In addition, the intersymbol interference (ISI) could degrade the signal causing a high bit error rate which may render an error correction engine useless.

SUMMARY

In one general aspect, a filter is created by sampling noise during an inter-frame gap of a received signal, sampling a data frame preamble from within a data frame of the received signal, and computing filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

Implementations may include one or more of the following features. For example, the filter may be structured and arranged to filter received signals communicated across power lines.

The received signal may be synchronized to identify the data frame preamble. The received signal may be synchronized by sampling a randomly located portion of the received signal, generating a pre-filter based on the randomly located portion of the sampled received signal, applying the pre-filter to the received signal, and identifying the data frame preamble based on the received signal after applying the pre-filter.

The filter coefficients may be computed by generating noise filter coefficients from the noise sampled during the inter-frame gap, generating channel filter coefficients from the data frame preamble sampled from within the data frame, and generating the filter coefficients based on the noise filter coefficients and the channel filter coefficients. The noise filter coefficients may be generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

The channel filter coefficients may be generated by identifying channel filter coefficients based on a threshold criteria and updating the channel filter coefficients that are identified based on the threshold criteria. The channel filter coefficients may be updated by performing an excision algorithm. The channel filter coefficients may be updated by modifying the channel filter coefficients. The channel filter coefficients may be modified by reducing a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

Additionally or alternatively, the channel filter coefficients may be updated by performing a smoothing algorithm. The channel filter coefficients may be updated by replacing the channel filter coefficients with replacement channel filter coefficients. In one implementation, the replacement channel filter coefficients may include channel filter coefficients not identified based on the threshold criteria.

In another general aspect, adaptively filtering a data frame of a received signal includes generating coefficients for an adaptive filter based on at least noise from within an inter-frame gap and a preamble of the data frame and filtering the data frame by applying the adaptive filter to the data frame.

Implementations may includes one or more of the following features. For example, the data frame may be communicated across power lines such that the data frame may be filtered by applying the adaptive filter to the data frame that is communicated across the power lines.

The coefficients for the adaptive filter may be generated by sampling noise during an inter-frame gap of a received signal, sampling a data frame preamble from within a data frame of the received signal, and computing filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

The received signal may be synchronized to identify the data frame preamble. The received signal may be synchronized by sampling a randomly located portion of the received signal, generating a pre-filter based on the randomly located portion of the sampled received signal, applying the pre-filter to the received signal, and identifying the data frame preamble based on the received signal after applying the pre-filter.

The filter coefficients may be computed by generating noise filter coefficients from the noise sampled during the inter-frame gap, generating channel filter coefficients from the data frame preamble sampled from within the data frame, and generating the filter coefficients based on the noise filter coefficients and the channel filter coefficients. The noise filter coefficients may be generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

The channel filter coefficients may be generated by identifying channel filter coefficients based on a threshold criteria and updating the channel filter coefficients that are identified based on the threshold criteria. The channel filter coefficients may be updated by performing an excision algorithm. The channel filter coefficients may be updated by modifying the channel filter coefficients. The channel filter coefficients may be modified by reducing a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

Additionally or alternatively, the channel filter coefficients may be updated by performing a smoothing algorithm. The channel filter coefficients may be updated by replacing the channel filter coefficients with replacement channel filter coefficients. In one implementation, the replacement channel filter coefficients may include channel filter coefficients not identified based on the threshold criteria.

In another general aspect, a filter is derived from a combination of coefficients. The coefficients used to derive the filter include coefficients derived from noise sampled during a period between data frames and a received data frame preamble and coefficients derived from the received data frame preamble and a transmitted data frame preamble.

Implementations may include one or more of the following features. For example, the coefficients derived from the noise may include an average of the coefficients derived from the noise over a period of time. The coefficients derived from the received data frame preamble may include an average of the coefficients derived from the data frame preamble over a period of time. The filter derived from the combination of the coefficients may include an average of the filters derived from the combination of the coefficients over a period of time. The coefficients derived from the noise may include an average of the noise over a period of time. The coefficients derived from the received data frame preamble may include an average of the data frame preamble over a period of time.

In another general aspect, receiving a signal transmitted through at least first and second communication paths over a single communication medium includes sampling the signal over the communication paths to realize a first sampling from the first communication path and a second sampling from the second communication path, synchronizing the first sampling and the second sampling, and combining the first sampling and the second sampling to generate a signal representative of the signal transmitted through the first and second communication paths.

Implementations may include one or more of the following features. For example, the first sampling and the second sampling may be synchronized by independently synchronizing the first sampling and the second sampling and adjusting for a delay difference between the first and the second communication paths based on the independent synchronization of the first sampling and the second sampling. The first communication path may include line-neutral path. The second communication path may include a neutral-ground path.

In another general aspect, pre-filtering a received signal to improve synchronization includes sampling a randomly located portion of the received signal, generating a pre-filter based on the randomly located portion of the sampled received signal, applying the pre-filter to the received signal, and identifying a data frame preamble based on the received signal after applying the pre-filter to improve synchronization.

Implementations may include one or more of the following features. For example, sampling the randomly located portion of the received signal may include sampling within noise and generating the pre-filter may include generating the pre-filter based on the sampled noise. The received signal may include data and noise. A portion of the received signal may be sampled at a randomly selected location.

In one implementation, the pre-filter may include a hybrid prediction filter that produces a desired response as an inverse of a value related to a portion of the received signal.

These general and specific aspects may be implemented using a system, a method, or a computer program, or any combination of systems, methods, or computer programs.

Other features will be apparent from the description and drawings, and from the claims.

DESCRIPTION OF DRAWINGS

Fig. 1 is a block diagram of a data frame including time intervals before and after the data frame.

Fig. 2 is a flow chart of an exemplary process for digital equalization.

Fig. 3 is a flow chart of an exemplary process for synchronizing an incoming bit stream.

Fig. 4 is a flow chart of an exemplary process for constructing a frequency-domain equalization and noise filter.

Fig. 5 is a flow chart of an exemplary process for computing a channel filter.

Fig. 6 is a block diagram of a system capable of performing digital equalization.

Fig. 7 is a block diagram of a system capable of performing data filtering.

Fig. 8 is a block diagram of a system capable of performing pre-filtering to enhance synchronization.

Fig. 9 is a block diagram of a system capable of improving reception of data frames under multipath propagation.

Like reference symbols in the various drawings may indicate like elements.

DETAILED DESCRIPTION

Fig. 1 illustrates a portion 100 of an exemplary data frame 105 including time intervals before and after the frame, such as an Inter-Frame Gap (IFG) 110. Data frame 105 may include bits transmitted over a communications system, including a preamble 115 and a data portion 125. The preamble 115 may include a synchronization preamble having a number of bits that are at the beginning of each data frame 105. Additionally or alternatively, preamble 115 may be located at the beginning of a data block, which may include several data portions 125 arranged in series (now shown). The preamble 115 may be used by a receiver to lock and to synchronize with the bit timing of a transmitter. A received signal may include any data, data gaps, and noise received, such as, for example, data frame 105, IFG 110, noise 120, preamble 115, and data 125.

IFG 110 may include a gap between data frames 105 in which data 125 typically is not transmitted. IFG 110 may include noise 120, e.g., resulting from a time interval with no transmitted data.

In general, a network system that transmits and receives data communications may detect the data frame 105 to perform processes such as, for example, digital equalization and filtering. For example, it may be desirable to construct noise filters and to instantaneously adapt the equalization and noise filters on a frame-by-frame basis. By instantaneously adapting the equalization and the noise filters on a frame-by-frame basis, the need for memory of the filter outside of the current data frame 105 and dependency on the previous data frame 105 may be reduced or eliminated altogether.

In one implementation, frequency-domain equalization may be used to retrieve the transmitted data over a noisy transmission channel with frequency-dependent attenuation (e.g., impairments of white noise, periodic noise, multipath propagation, and different impedances). A filter may be designed to reduce or eliminate the noise, correct for the channel distortion and the channel attenuation, and/or compensate for the ISI.

In this implementation, the desired equalization and filtering may be accomplished by using the IFG 110 to sample the noise, and then use the captured noise and the preamble 115 to construct a filter (e.g., an optimal Wiener filter).

Fig. 2 illustrates a process 200 for digital equalization. Process 200 typically includes synchronizing an incoming data frame (step 210), constructing a frequency-domain equalization and noise filter (step 220), and applying the filter to the frame from which it was derived/constructed (step 230).

In general, synchronizing the incoming frame (step 210) includes receiving a signal and locating a frame, namely, identifying the beginning and/or the end of the frame. Various modulation schemes may be used in transmitting and receiving bit streams. In one example implementation, a binary phase shift keying (BPSK) modulation scheme may be used, however, other modulation schemes are also available for application to the digital equalization process.

For synchronization (step 210), the incoming signal may be over-sampled (e.g., by a factor of N) and an algorithm (e.g., a maximal likelihood algorithm) may be used to estimate the incoming bit value. The received bit sequence (e.g., a 64-bit preamble) then may be correlated to the predefined and stored transmitted bit sequence and submitted to decision logic that may determine if the sequence corresponds to the predefined transmitted preamble. The synchronization process 210 may be performed at the receiving location of the incoming bit stream. For example, the synchronization 210 may occur at a receiver circuit located at the receiving location. Synchronization process 210 may be used to maintain proper bit timing at the receiver location. Synchronization process 210 may involve locating a sync position (e.g., locating the preamble). By identifying the beginning and/or the end of the frame in the synchronization (step 210), the IFG may be located and noise within the IFG may be sampled in step 220.

Constructing the frequency-domain equalization and noise filter (step 220) may include using one or more samples of the noise in the IFG, the received preamble, and the transmitted preamble. Once the incoming frame has been synchronized (step 210), the noise may be sampled and one or more filters may be computed and constructed. For instance, two complementary filters may be computed (e.g., a noise filter and a channel filter). A combination of the noise filter and the channel filter may be used to generate a filter, such as a Wiener filter or some other filter capable of being used as an optimal filter. Additionally and/or alternatively, the construction of the filter (step 220) may include using an average of samples of the noise in the IFG, the received preamble, the transmitted preamble, and/or constructed filters over a period of time.

Applying the filter to the frame from which it was derived (step 230) may include multiplying the filter coefficients by the incoming data frame. Examples of the processes described by steps 210 and 220 are described below in more detail with respect to Figs. 3 and 4-5, respectively.

Fig. 3 is an expansion of the synchronization process 210 from Fig. 2. Fig. 3 typically includes sampling the incoming signal (step 310), estimating the incoming bit value (step 320), and correlating the received bit sequence (e.g., the 64-bit preamble) to a predefined bit sequence that was transmitted as the preamble (step 330).

Over-sampling the incoming signal (step 310) may include over-sampling the incoming signal by a factor of N. Over-sampling of the signal (step 310) may be performed by an analog-to-digital converter (ADC), or by another circuit capable of over-sampling an incoming signal. Estimating the incoming bit value (step 320) may include demodulating the over-sampled incoming data into a single binary bit stream. Correlating the received bit sequence (step 330) may include correlating the over-sampled incoming data stream against a stored copy of the transmitted bit wave form. For example, in step 330, the incoming data stream may be correlated against a sine wave. For a more detailed description of the synchronization process 210, section 5.1 Synchronizer is provided below, which includes subsections 5.1.1 Demodulator, 5.1.1.1 Correlator, 5.1.1.2 Threshold Function, and 5.1.2 Synchronization.

Fig. 4 illustrates one implementation of the process 220 of Fig. 2 for constructing a frequency-domain equalization and noise filter. Constructing the filter typically includes identifying noise samples, the received preamble, and predefined transmitted preamble (step 410), computing a noise filter (step 420), a channel filter (step 430), and ultimately a filter that is applied to the data frame (step 440). For a more detailed description of the filter construction process, section 5.2 Filter is provided below with subsections 5.2.1 Noise Filter, 5.2.1.1 Noise Filter Computation, 5.2.2 Channel Filter, 5.2.2.1 Raw Channel Filter Computation, 5.2.2.2 Excision, and 5.2.2.3 Smoothing.

A channel filter may be computed (step 430) as described more fully with respect to Fig. 5. Computing a channel filter typically includes computing the raw coefficients (step 510), performing excision (step 520) and performing smoothing (step 530). For a more detailed description of the channel filter construction process, sections 5.2.2.1 Raw Channel Filter Computation, 5.2.2.2 Excision, and 5.2.2.3 Smoothing are provided below.

Fig. 6 illustrates a system for digital equalization. The digital equalization system typically receives an incoming signal 610 (e.g., an analog signal) and applies a receiver/over-sampler module (e.g., an analog-to-digital converter) 620, a synchronizer 630, a demodulator module 640, a synchronization module 650, and a filter coefficient module 660.

5.1 Synchronizer 630

The synchronizer 630 generally is capable of detecting the preamble of the incoming data. The synchronizer 630 typically includes two components: a demodulation/ correlation module (maximum likelihood at the bit level) and a synchronization module 650 (maximum likelihood for the preamble).

5.1.1 Demodulator Module 640

The demodulator module 640 generally is capable of converting over-sampled incoming data into a single binary bitstream of "ones" and "zeroes" (i.e., BitOut). The demodulator module 640 typically includes a correlator 641 and a threshold detector 643.

5.1.1.1 Correlator 641

The sampled data stream from the analog-to-digital converter (ADC) 620 goes into the correlator 641. For each new sample S_j, N samples are correlated against a stored copy of the transmitted bit waveform 642 (e.g., in one implementation, a sine wave is used). Thus, for comparison, the transmitted bit waveform 642 is effectively moved across the input data stream like a sliding window, for example, effecting the logic of Equations 1 and 2:

$$BitWeight = \sum_{i=0}^{N-1} receivedBit[i] \times transmittedBit[i] \qquad \text{(Equation 1)}$$

with

$$transmittedBit[n] = \begin{cases} \sin(2\pi n / N) & for transmitting\ 1 \\ -\sin(2\pi n / N) & for transmitting\ 0 \end{cases} \qquad n = 0 \cdots N-1$$

(Equation 2)

and where N is the over-sampling rate.

The BitWeight generated by the correlator 641 generally has a new value for every sample S_j; it is forwarded to a decision logic (the threshold detector 643) that determines if the transmitted bit is "0" or "1" based on the correlation of the sampled region to the pre-stored transmitted bit preamble as reflected by the BitWeight.

5.1.1.2 Threshold Function

The resultant BitWeight is passed to the threshold detector 643 to be processed. The values are compared to defined max and min thresholds (both defaulted to "0").

The threshold detector 643 will convert each BitWeight value into one logical data bit value. This process is shown in the following equation:

```
If ($BitWeight$ > max_threshold)
        $BitOut = 1$;                //the received bit is a "1"
If ($BitWeight$ < min_threshold)
        $BitOut = 0$;                //the received bit is a "0"
Else
        $BitOut$ = random generated 0 or 1 //no decision can be made
```

(Equation 3)

5.1.2 Synchronization Module 650

The synchronization module 650 detects the beginning and/or the end of the frame. For each new sample s_j, the receiver 651 receives the BitOut sequence and the comparing device 653 computes the *Hamming distance*, d_j, between the received symbol sequence (received preamble) and the transmitted symbol sequence (transmitted preamble). The smallest value of d_j for all samples s_j is called the *minimum distance* $d_{minfolding}$, which may be used to indicate the frame position (e.g., a maximum likelihood detection). The position j for which the Hamming distance is lowest generally is considered the sync position.

The value of the Hamming distance may be passed to the folding function module 654 which may convert the Hamming distance from a [0…64] range number (e.g., a number corresponding to the number of bits in the preamble) into "Folded Hamming distance" which includes a [0…32] range number according to the following equation:

```
if Hamming distance > 32 then
        Folded Hamming distance = 64 – Hamming distance
else
        Folded Hamming distance = Hamming distance.
```

The folding function module 654 enables the detection of both positive and negative phases of the transmitted signal. For example, a phase inversion may occur if the transmitter or receiver is incorrectly plugged into a power outlet. In this instance, the transmitted bits may be inverted in sign causing a "match" condition to correspond to maximum Hamming distance values instead of the minimum Hamming distance values. By folding the Hamming distance, both positive and negative phases of the transmitted signal may be detected by examining the minimum Hamming distance values. The position for which the Folded Hamming distance is minimal $d_{minfolding}$ may be considered as the sync position.

5.2 Filter Coefficient Module 660

The Filter Coefficient Module 660 generates frequency-domain coefficients that will be used to filter the received data stream that are useful in recovering the information content of the transmitted data. In one implementation, for example, the filter coefficient module 660 also may be implemented using an Adaptive Filter Coefficient Engine (AFCE).

In one implementation, the coefficient generation includes determination of channel, noise filters, and ultimately the complex multiplication (i.e., frequency-domain operation) of the channel and noise filters, as described below with respect to Equation 4.

$$\text{FilterCoeffs}(f) = \text{ChannelFilter}(f) \times \text{NoiseFilter}(f) \qquad \text{(Equation 4)}$$

5.2.1 Noise filter 661

The noise filter generation module 661 may include a filter capable of minimizing the effect of the noise generated by the transmission channel. The noise filter generation module 661 generally is based, at least in part, on a sampling of noise from within the quiet period where no data is transmitted, which generally precedes the preamble. It also may be based on data within the preamble identified through synchronization. The sampled noise and/or preamble generally are combined to construct the noise filter itself. This approach is valid as long as the frame length is short enough that the noise could be considered as quasistationary for the frame duration. For the purposes of this example implementation, it may be assumed that the noise is uncorrelated to the data and that the noise is considered quasistationary for the frame duration.

5.2.1.1 Noise Filter Computation

The noise filter may be computed based on equation 5:

$$\text{NoiseFilter(f)} = \frac{|\text{received preamble(f)}|^2 - |\text{noise(f)}|^2}{|\text{received preamble(f)}|^2} \qquad \text{(Equation 5)}$$

where $|\text{received preamble(f)}|^2$ represents the power of individual frequency components comprising the discrete spectrum for the received preamble and where $|\text{noise(f)}|^2$ may be determined using equation 6:

$$|\text{noise(f)}|^2 = \min(|\text{noiseRx\_1(f)}|^2, |\text{noiseRx\_2(f)}|,^2 \ldots, |\text{noiseRx\_k(f)}|^2) \qquad \text{(Equation 6)}$$

where $|\text{noiseRx\_1(f)}|^2$, $|\text{noiseRx\_2(f)}|^2$, …, $|\text{noiseRx\_k(f)}|^2$ are obtained based on noise snapshots from within the IFG, and k is the number of noise blocks that are captured. As equation 6 is based on the minimum of the power of the individual noise frequency components of the noise streams, it represents a conservative approach (i.e., versus using an average) to characterizing the noise for the noise filter computation. If k is too large, the $|\text{noise(f)}|^2$ component may tend toward "0" for all frequencies counteracting the effect of this conservative characterization of noise. As such, k generally is set to a relatively small value, e.g., 3.

5.2.2 Channel filter 662

The channel filter 662 may be a filter capable of characterizing the channel and compensating for the channel attenuation and distortion. The channel may be characterized by comparing the received preamble spectrum against the transmitted preamble spectrum. Then, the raw channel filter may be computed by dividing the spectrum of the transmitted preamble bit sequence by the spectrum of the received preamble bit sequence.

Once the raw channel filter has been obtained, additional steps/processing may be performed to eliminate additional interference (e.g., the narrow band interference) and to increase the accuracy of the estimate in the presence of noise.

5.2.2.1 Raw Channel Filter Computation Module 6621

One possible equation for computing the raw channel filter is described as follows. The channel filter 662 may be computed based on frequency-domain division (i.e. complex number division) of the prestored/predetermined transmitter preamble by the received/detected preamble. This is illustrated in the following equation:

$$\text{RawChannelFilter(f)} = \frac{\text{prestored/predetermined transmitted preamble(f)}}{\text{received/detected preamble(f)}} \quad \text{(Equation 7)}$$

For instance, the raw channel filter may be modified by post-processing functions identified as "excision" 6622 and "smoothing" 6623. The excision processing 6622 reduces the gross anomalies of the spectrum, while the smoothing processing 6623 increases the accuracy of the estimate in the presence of noise.

In the implementation described below, excision is performed before smoothing. However, either could be performed independently, or the order could be reversed.

5.2.2.2 Excision Module 6622

Excision 6622 may be used to significantly improve the signal-to-noise ratio by eliminating the narrow-band interference at the receiver. Several excision algorithms generally are known to those of ordinary skill, as have been described. See Analysis of DFT-Based Frequency Excision Algorithms for Direct-Sequence Spread-Spectrum Communications. IEEE Transactions on Communications, vol. 46, No. 8, August 1998. Jeffrey A. Young, and James S. Lehnert.

When using the excision algorithm, the frequency bin(s) to be excised is/are generally identified according to their relative ranking under a predetermined ranking scheme, and the magnitude and method of excision or notching is determined.

In one implementation, the frequency bins having a power value that places them within a selectively high percentage (e.g., the top M percent) of the frequency bins are identified for excision, during which the power values for those frequency bins are reduced by a selectable amount or ratio (e.g., half). More specifically, according to one exemplary implementation of the excision module 6622, power values for the raw channel filter coefficients 6621 are calculated and sorted, and the top M percent of the raw channel filter coefficients are selected and divided by two. An example implementation is illustrated in the following equation:

if $(RawChannelFilter(f_i) * noiseFilter(f_i))$ in top M% then

$$ExcisedChannelFilter(f_i) = ExcisedChannelFilter(f_i)/2$$

Else

$$ExcisedChannelFilter(f_i) = ExcisedChannelFilter(f_i)$$

$$i = 0 \cdots size\,of\ the\,FFT\ filter - 1$$

In this implementation, the excised channel filter spectrum is subject to a smoothing function following the excision.

5.2.2.3 Smoothing Module 6623

The smoothing function performed by smoothing module 6623 can be summarized by the following logic:

if the power of the $|ExcisedChannelFilter(f_i)|^2$ component is greater than the power of the $|ExcisedChannelFilter(f_{i+1})|^2$ component, then the final channel filter for bin "i" will be assigned the $ExcisedChannelFilter(f_{i+1})$ bin value.

An algorithm implementing this logic is as follows:

```
$if|ExcisedChannelFilter(f_i)|^2 \le |ExcisedChannelFilter(f_{i+1})|^2$
    $channelFilter(f_i) = ExcisedChannelFilter(f_i)$
$else$
    $channelFilter(f_i) = ExcisedChannelFilter(f_{i+1})$
```

where f_i cover both the negative and the positive spectrum of the signal.

In one implementation, following calculation of the noise filter coefficients and the channel filter coefficients, a multiplier 663 may be used to generate the ultimate coefficients by performing complex multiplication of the channel and noise filter coefficients, as described above with respect to equation 4, such that a filter may be constructed using the filter computation module 670.

5.2.3 Data Filtering

Fig. 7 illustrates an overview of the system 700 used to generate and apply a filter. In one implementation, the overlap-save method may be used to filter the data. The overlap–save method performs a linear convolution between a finite-length sequence (the channel filter coefficients) and an infinite-length sequence (the data) by appropriately partitioning the data. This type of data filtering method generally includes an input module 710, which receives the input data x(n). The input module 710 may concatenate new input data x(n) with old input data. The concatenated data may be sent to a first Fast Fourier Transform (FFT) module 720, where it is used to produce an output X(f).

Component parts of the input data x(n) (e.g., a preamble (n) and noise (n)) may be received at a filter coefficient computation module 730. The filter coefficient computation module 730 also may receive, as input, a copy of the transmitted preamble 740. The filter coefficient computation module 730 computes the appropriate filters and sends its output to a second FFT module 750, which outputs coeff (f). The output X(f) from the first FFT module 720 is multiplied with the output coeff (f) from the second FFT module 750 at multiplier module 760.

The output of multiplier module 760 Y(f) is received at an Inverse FFT module 770, where an inverse FFT process may be performed. The result of the inverse FFT process is saved in a buffer module 780 and the output filtered data y(n) is produced.

Additional improvement(s) and a lower Bit Error Rate may be obtained by pre-filtering the incoming raw data and/or adding additional hardware that further solves the multipath propagation problem.

6.1 Pre-Filtering

Pre-filtering may be performed independently and exclusively, or, in conjunction with of any other filtering processes, such as the filtering processes described above.

As described in section 5.2, the filter coefficient computation may be performed by constructing a channel inverse. This computation generally requires synchronization on the raw signal. Indeed, if the signal is jammed or severely corrupted by noise (e.g., white noise, periodic noise, etc.), the synchronizer could fail and the frame synchronization may not be detectable.

In one implementation, the raw data may be pre-filtered to improve the synchronization capability and thus improve the filter/equalizer performance (section 5.2). Indeed, since the filter/equalizer process uses the raw received preamble to compute the channel inverse, the computed filter coefficients will generally improve with improvements to estimates of the preamble position (the sync position).

More specifically, to pre-filter, the dead time between the blocks may be used to sample the noise and to construct an adaptive noise-canceling filter that is helpful in making preamble bits more identifiable.

Several noise filters could be used for this purpose. For instance, in one implementation, a noise filter used for pre-filtering includes a hybrid version of a "Prediction" filter, where the desired response for the adaptive filter is the inverse of the present input signal value.

Fig. 8 illustrates an example implementation of a pre-filtering system. The pre-filtering system of Fig. 8 receives an input signal u(t) that is received at delay module 810, and includes an adaptive filter module 820 and a summation module 830.

In this implementation, the following notations are used for convenience:

s = transmitted signal (uncorrupted by noise),

n = channel noise (white noise, periodic noise...),

$\tilde{n}$ = estimate of the channel noise,

u = input applied to the adaptive filter,

y = output of the adaptive filter,

d = desired response of the filter,

$e = -(y + d)$ = estimation error.

This adaptive filter 820 predicts the inverse of the present value of the random input signal, even though past values of the signal supply the input applied to the adaptive filter. The present value of the signal serves the purpose of the desired response for the adaptive filter 820.

When the filter adaptation algorithm (e.g., least mean square algorithm) is enabled, the adaptation process is designed to cancel the input signal $u(t)$ by adjusting the filter coefficients such that $y(t) = u(t) + \textit{the filter output} = \text{"}0\text{"}$. Assuming that during this adaptation process, $u(t)$ was selected such that it is equal to the noise in the transmission channel (i.e. $u(t) = n(t)$ = the sampled noise during the dead time between the frame(s)), the adaptation process will basically construct a noise-canceling filter that cancels the transmission channel noise such that $y(t) = n(t) - \tilde{n}(t) \cong 0$.

Once the adaptation process is complete, the adaptation algorithm may be disabled and the obtained filter coefficients locked. Assuming that the noise is quasistationary for the frame duration (see assumption above), the obtained filter could then be considered as a noise-canceling filter that is continuously predicting the inverse of the noise in the transmission channel $y(t) = n(t) - \tilde{n}(t) \cong 0$. Thus, when data is transmitted $u(t) = s(t) + n(t)$ the filter will cancel the noise $n(t)$ and let pass the transmitted data such that $y(t) = s(t) + n(t) - \tilde{n}(t) \cong s(t)$.

In another implementation of a pre-filter, for example, if during the period of the preamble, $u(t)$ is selected such that it is equal to the transmitted preamble, then the adaptation process may construct an equalization filter.

6.2 Multipath Propagation

In one implementation, a modem may be designed to transmit on two paths (e.g., a first may be a Line-Neutral path and a second may be a Neutral-Ground path). In theory, the transmitted data (e.g., equalized signals) could be recovered by sampling either path. However, the two paths may have different characteristics due to the difference in the path loads (time-varying values), impedance differences, length differences, and/or noise levels. This frequency-selective fading of the transmission medium could cause amplitude and delay distortion which may degrade the system reliability beyond that expected. It therefore may be difficult to determine which path is the best one to use to recover the transmitted data, whether it is the first path, the second path, or a combination of both paths.

Fig. 9 illustrates one example implementation of a multipath propagation system capable of automatically determining which path or combination of paths is preferred, or optimal. The system may include structures that automatically probe/sample the paths (e.g., Line-Neutral 910 and Neutral-Ground 920), store the data (e.g., by using one or more buffers 930a, 930b), and synchronize (e.g., using synchronizers 940a, 940b) on each path, independently. Further, once a synchronizer 940a, 940b has been detected, the system automatically adjusts for the delay difference between the paths by adjusting the read pointer in the buffers 930a, 930b and then combines the data into a single data stream following gain adjustment module 950.

Indeed, assuming that the buffers are large enough, if we are able to synchronize on both paths 910, 920 (discussed above in section 5.1), the difference in the sync position typically is equal to the difference in the path delays. Thus, the solution presents itself in two parts: first, the synchronizers may be used to compensate for the path delay differences, and second, the filter/equalizer discussed above in section 5.2 may be used to compensate for the distortion. It will be understood that various modifications may be made without departing from the spirit and scope of the claims. For example, the order of the operations (path selection and filtering/equalization) could be changed. Indeed, the data filtering/equalization part could be done on each path independently then data could be combined.

If the attenuation/distortion on either path is too severe such that no synchronization is possible, a single path solution may be realized without any path combination. In another implementation, the addition of "combination thresholds" may refine this feature even further by enabling the two paths to be combined if and only if the quality of the received data on the paths is above a predefined level.

The described systems, methods, and techniques may be implemented in digital electronic and/or analog circuitry, computer hardware, firmware, software, or in combinations of these elements. Apparatus embodying these techniques may include appropriate input and output devices, a computer processor, and a computer program product tangibly embodied in a machine-readable storage device for execution by a programmable processor. A process embodying these techniques may be performed by a programmable processor executing a program of instructions to perform desired functions by operating on input data and generating appropriate output. The techniques may be implemented in one or more computer programs that are executable on a programmable system including at least one programmable processor coupled to receive data and instructions from, and to transmit data and instructions to, a data storage system, at least one input device, and at least one output device. Each computer program may be implemented in a high-level procedural or object-oriented programming language, or in assembly or machine language if desired; and in any case, the language may be a compiled or interpreted language. Suitable processors include, by way of example, both general and special purpose microprocessors. Generally, a processor will receive instructions and data from a read-only memory and/or a random access memory. Storage devices suitable for tangibly embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, such as Erasable Programmable Read-Only Memory (EPROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and flash memory devices; magnetic disks such as internal hard disks and removable disks; magneto-optical disks; and Compact Disc Read-Only Memory (CD-ROM). Any of the foregoing may be supplemented by, or incorporated in, specially-designed ASICs (application-specific integrated circuits).

Advantageous results still could be achieved if steps of the disclosed techniques were performed in a different order and/or if components in the disclosed systems were combined in a different manner and/or replaced or supplemented by other components. For example, the transmission medium may include a power line, a telephone line, a cable line, a digital subscriber line (DSL), an integrated services digital network (ISDN) line, a radio frequency (RF) medium, and/or other transmission media. Additionally and/or alternatively, different modulation schemes are also available for application to the digital equalization process. For example, a combination of modulation schemes may be applied to different parts of the signal (e.g., one modulation scheme may be applied to the preamble and a different modulation scheme may be applied to the remainder of the signal). In another implementation, for example, in a multipath propagation implementation, different modulation schemes and/or combinations of different modulation schemes may be applied to each signal on the different communication paths. Accordingly, other implementations are within the scope of the following claims.

WHAT IS CLAIMED IS:

1. A method for creating a filter, the method comprising:

sampling noise during an inter-frame gap of a received signal;

sampling a data frame preamble from within a data frame of the received signal; and

computing filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

2. The method as in claim 1 wherein the filter is structured and arranged to filter received signals communicated across power lines.

3. The method as in claim 1 further comprising synchronizing the received signal to identify the data frame preamble.

4. The method as in claim 3 wherein synchronizing the received signal comprises:

sampling a randomly located portion of the received signal;

generating a pre-filter based on the randomly located portion of the sampled received signal;

applying the pre-filter to the received signal; and

identifying the data frame preamble based on the received signal after applying the pre-filter.

5. The method as in claim 1 wherein computing the filter coefficients includes:

generating noise filter coefficients from the noise sampled during the inter-frame gap;

generating channel filter coefficients from the data frame preamble sampled from within the data frame; and

generating the filter coefficients based on the noise filter coefficients and the channel filter coefficients.

6. The method as in claim 5 wherein the noise filter coefficients are generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

7. The method as in claim 5 wherein generating the channel filter coefficients includes:

identifying channel filter coefficients based on a threshold criteria; and

updating the channel filter coefficients that are identified based on the threshold criteria.

8. The method as in claim 7 wherein updating the channel filter coefficients includes performing an excision algorithm.

9. The method as in claim 7 wherein updating the channel filter coefficients includes modifying the channel filter coefficients.

10. The method as in claim 9 wherein modifying the channel filter coefficients includes reducing a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

11. The method as in claim 7 wherein updating the channel filter coefficients includes performing a smoothing algorithm.

12. The method as in claim 7 wherein updating the channel filter coefficients includes replacing the channel filter coefficients with replacement channel filter coefficients.

13. The method as in claim 12 wherein the replacement channel filter coefficients include channel filter coefficients not identified based on the threshold criteria.

14. A system for creating a filter, comprising:

means for sampling noise during an inter-frame gap of a received signal;

means for sampling a data frame preamble from within a data frame of the received signal; and means for computing filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

15. The system of claim 14 wherein the filter is structured and arranged to filter received signals communicated across power lines.

16. The system of claim 14 further comprising means for synchronizing the received signal to identify the data frame preamble.

17. The system of claim 16 wherein the means for synchronizing the received signal comprises:

means for sampling a randomly located portion of the received signal;

means for generating a pre-filter based on the randomly located portion of the sampled received signal;

means for applying the pre-filter to the received signal; and

means for identifying the data frame preamble based on the received signal after applying the pre-filter.

18. The system of claim 14 wherein the means for computing the filter coefficients includes:

means for generating noise filter coefficients from the noise sampled during the inter-frame gap;

means for generating channel filter coefficients from the data frame preamble sampled from within the data frame; and

means for generating the filter coefficients based on the noise filter coefficients and the channel filter coefficients.

19. The system of claim 18 wherein the noise filter coefficients are generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

20. The system of claim 18 wherein the means for generating the channel filter coefficients includes:

means for identifying channel filter coefficients based on a threshold criteria; and

means for updating the channel filter coefficients that are identified based on the threshold criteria.

21. The system of claim 20 wherein the means for updating the channel filter coefficients includes means for performing an excision algorithm.

22. The system of claim 20 wherein the means for updating the channel filter coefficients includes means for modifying the channel filter coefficients.

23. The system of claim 22 wherein the means for modifying the channel filter coefficients includes means for reducing a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

24. The system of claim 20 wherein the means for updating the channel filter coefficients includes means for performing a smoothing algorithm.

25. The system of claim 20 wherein the means for updating the channel filter coefficients includes means for replacing the channel filter coefficients with replacement channel filter coefficients.

26. The system of claim 25 wherein the replacement channel filter coefficients include channel filter coefficients not identified based on the threshold criteria.

27. A computer program stored on a computer readable medium or a propagated signal for creating a filter, comprising:

a sampling code segment that causes the computer to sample noise during an inter-frame gap of a received signal and to sample a data frame preamble from within a data frame of the received signal; and

a computing code segment that causes the computer to compute filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

28. The computer program of claim 27 wherein the filter is structured and arranged to filter received signals communicated across power lines.

29. The computer program of claim 27 further comprising a synchronizing code segment that causes the computer to synchronize the received signal to identify the data frame preamble.

30. The computer program of claim 29 wherein the synchronizing code segment includes:

a sampling code segment that causes the computer to sample a randomly located portion of the received signal;

a generating code segment that causes the computer to generate a pre-filter based on the randomly located portion of the sampled received signal;

an applying code segment that causes the computer to apply the pre-filter to the received signal; and

an identifying code segment that causes the computer to identify the data frame preamble based on the received signal after applying the pre-filter.

31. The computer program of claim 27 wherein the computing code segment includes:

a first generating code segment that causes the computer to generate noise filter coefficients from the noise sampled during the inter-frame gap;

a second generating code segment that causes the computer to generate channel filter coefficients from the data frame preamble sampled from within the data frame; and

a third generating code segment that causes the computer to generate the filter coefficients based on the noise filter coefficients and the channel filter coefficients.

32. The computer program of claim 31 wherein the noise filter coefficients are generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

33. The computer program of claim 31 wherein the third generating code segment includes:

an identifying code segment that causes the computer to identify channel filter coefficients based on a threshold criteria; and

an updating code segment that causes the computer to update the channel filter coefficients that are identified based on the threshold criteria.

34. The computer program of claim 33 wherein the updating code segment includes a performing code segment that causes the computer to perform an excision algorithm.

35. The computer program of claim 33 wherein the updating code segment includes a modifying code segment that causes the computer to modify the channel filter coefficients.

36. The computer program of claim 35 wherein the modifying the code segment includes a reducing code segment that causes the computer to reduce a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

37. The computer program of claim 33 wherein the updating code segment includes a performing code segment that causes the computer to perform a smoothing algorithm.

38. The computer program of claim 33 wherein the updating code segment includes a replacing code segment that causes the computer to replace the channel filter coefficients with replacement channel filter coefficients.

39. The computer program of claim 38 wherein the replacement channel filter coefficients include channel filter coefficients not identified based on the threshold criteria.

40. A method for adaptively filtering a data frame of a received signal, the method comprising:

generating coefficients for an adaptive filter based on at least noise from within an inter-frame gap and a preamble of the data frame; and filtering the data frame by applying the adaptive filter to the data frame.

41. The method as in claim 40 wherein the data frame is communicated across power lines such that filtering the data frame includes applying the adaptive filter to the data frame communicated across the power lines.

42. The method as in claim 40 wherein generating the coefficients for the adaptive filter includes:

sampling noise during an inter-frame gap of a received signal;

sampling a data frame preamble from within a data frame of the received signal; and

computing filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

43. The method as in claim 42 further comprising synchronizing the received signal to identify the data frame preamble.

44. The method as in claim 43 wherein synchronizing the received signal comprises:

sampling a randomly located portion of the received signal;

generating a pre-filter based on the randomly located portion of the sampled received signal;

applying the pre-filter to the received signal; and

identifying the data frame preamble based on the received signal after applying the pre-filter.

45. The method as in claim 42 wherein computing the filter coefficients includes:

generating noise filter coefficients from the noise sampled during the inter-frame gap;

generating channel filter coefficients from the data frame preamble sampled from within the data frame; and generating the filter coefficients based on the noise filter coefficients and the channel filter coefficients.

46. The method as in claim 45 wherein the noise filter coefficients are generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

47. The method as in claim 45 wherein generating the channel filter coefficients includes:

identifying channel filter coefficients based on a threshold criteria; and

updating the channel filter coefficients that are identified based on the threshold criteria.

48. The method as in claim 47 wherein updating the channel filter coefficients includes performing an excision algorithm.

49. The method as in claim 47 wherein updating the channel filter coefficients includes modifying the channel filter coefficients.

50. The method as in claim 49 wherein modifying the channel filter coefficients includes reducing a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

51. The method as in claim 47 wherein updating the channel filter coefficients includes performing a smoothing algorithm.

52. The method as in claim 47 wherein updating the channel filter coefficients includes replacing the channel filter coefficients with replacement channel filter coefficients.

53. The method as in claim 52 wherein the replacement channel filter coefficients include channel filter coefficients not identified based on the threshold criteria.

54. A system for adaptively filtering a data frame of a received signal, comprising:

means for generating coefficients for an adaptive filter based on at least noise from within an inter-frame gap and a preamble of the data frame; and

means for filtering the data frame by applying the adaptive filter to the data frame.

55. The system of claim 54 wherein the data frame is communicated across power lines such that the means for filtering the data frame includes means for applying the adaptive filter to the data frame communicated across the power lines.

56. The system of claim 54 wherein the means for generating the coefficients for the adaptive filter includes:

means for sampling noise during an inter-frame gap of a received signal;

means for sampling a data frame preamble from within a data frame of the received signal; and

means for computing filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

57. The system of claim 56 wherein the filter is structured and arranged to filter received signals communicated across power lines.

58. The system of claim 56 further comprising means for synchronizing the received signal to identify the data frame preamble.

59. The system of claim 58 wherein the means for synchronizing the received signal comprises:

means for sampling a randomly located portion of the received signal;

means for generating a pre-filter based on the randomly located portion of the sampled received signal;

means for applying the pre-filter to the received signal; and

means for identifying the data frame preamble based on the received signal after applying the pre-filter.

60. The system of claim 56 wherein the means for computing the filter coefficients includes:

means for generating noise filter coefficients from the noise sampled during the inter-frame gap;

means for generating channel filter coefficients from the data frame preamble sampled from within the data frame; and

means for generating the filter coefficients based on the noise filter coefficients and the channel filter coefficients.

61. The system of claim 60 wherein the noise filter coefficients are generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

62. The system of claim 60 wherein the means for generating the channel filter coefficients includes:

means for identifying channel filter coefficients based on a threshold criteria; and

means for updating the channel filter coefficients that are identified based on the threshold criteria.

63. The system of claim 62 wherein the means for updating the channel filter coefficients includes means for performing an excision algorithm.

64. The system of claim 62 wherein the means for updating the channel filter coefficients includes means for modifying the channel filter coefficients.

65. The system of claim 64 wherein the means for modifying the channel filter coefficients includes means for reducing a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

66. The system of claim 62 wherein the means for updating the channel filter coefficients includes means for performing a smoothing algorithm.

67. The system of claim 62 wherein the means for updating the channel filter coefficients includes means for replacing the channel filter coefficients with replacement channel filter coefficients.

68. The system of claim 67 wherein the replacement channel filter coefficients include channel filter coefficients not identified based on the threshold criteria.

69. A computer program stored on a computer readable medium or a propagated signal for adaptively filtering a data frame of a received signal, comprising:

a generating code segment that causes the computer to generate coefficients for an adaptive filter based on at least noise from within an inter-frame gap and a preamble of the data frame; and

a filtering code segment that causes the computer to filter the data frame by applying the adaptive filter to the data frame.

70. The computer program of claim 69 wherein the data frame is communicated across power lines such that the filtering code segment causes the computer to apply the adaptive filter to the data frame communicated across the power lines.

71. The computer program of claim 69 wherein the generating code segment includes:.

a sampling code segment that causes the computer to sample noise during an inter-frame gap of a received signal and to sample a data frame preamble from within a data frame of the received signal; and

a computing code segment that causes the computer to compute filter coefficients based on the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

72. The computer program of claim 71 wherein the filter is structured and arranged to filter received signals communicated across power lines.

73. The computer program of claim 71 further comprising a synchronizing code segment that causes the computer to synchronize the received signal to identify the data frame preamble.

74. The computer program of claim 73 wherein the synchronizing code segment includes:

a sampling code segment that causes the computer to sample a randomly located portion of the received signal;

a generating code segment that causes the computer to generate a pre-filter based on the randomly located portion of the sampled received signal;

an applying code segment that causes the computer to apply the pre-filter to the received signal; and

an identifying code segment that causes the computer to identify the data frame preamble based on the received signal after applying the pre-filter.

75. The computer program of claim 71 wherein the computing code segment includes:

a first generating code segment that causes the computer to generate noise filter coefficients from the noise sampled during the inter-frame gap;

a second generating code segment that causes the computer to generate channel filter coefficients from the data frame preamble sampled from within the data frame; and

a third generating code segment that causes the computer to generate the filter coefficients based on the noise filter coefficients and the channel filter coefficients.

76. The computer program of claim 75 wherein the noise filter coefficients are generated from the noise sampled during the inter-frame gap and the data frame preamble sampled from within the data frame.

77. The computer program of claim 75 wherein the third generating code segment includes:

an identifying code segment that causes the computer to identify channel filter coefficients based on a threshold criteria; and

an updating code segment that causes the computer to update the channel filter coefficients that are identified based on the threshold criteria.

78. The computer program of claim 77 wherein the updating code segment includes a performing code segment that causes the computer to perform an excision algorithm.

79. The computer program of claim 77 wherein the updating code segment includes a modifying code segment that causes the computer to modify the channel filter coefficients.

80. The computer program of claim 79 wherein the modifying the code segment includes a reducing code segment that causes the computer to reduce a magnitude of the channel filter coefficients that are identified based on the threshold criteria.

81. The computer program of claim 77 wherein the updating code segment includes a performing code segment that causes the computer to perform a smoothing algorithm.

82. The computer program of claim 77 wherein the updating code segment includes a replacing code segment that causes the computer to replace the channel filter coefficients with replacement channel filter coefficients.

83. The computer program of claim 82 wherein the replacement channel filter coefficients include channel filter coefficients not identified based on the threshold criteria.

84. A filter derived from a combination of coefficients, the coefficients used to derive the filter comprising:

coefficients derived from noise sampled during a period between data frames and a received data frame preamble; and

coefficients derived from the received data frame preamble and a transmitted data frame preamble.

85. The filter of claim 84 wherein the coefficients derived from the noise include an average of the coefficients derived from the noise over a period of time.

86. The filter of claim 84 wherein the coefficients derived from the received data frame preamble include an average of the coefficients derived from the data frame preamble over a period of time.

87. The filter of claim 84 wherein the filter derived from the combination of the coefficients includes an average of the filters derived from the combination of the coefficients over a period of time.

88. The filter of claim 84 wherein the coefficients derived from the noise include an average of the noise over a period of time.

89. The filter of claim 84 wherein the coefficients derived from the received data frame preamble include an average of the data frame preamble over a period of time.

90. A method for receiving a signal transmitted through at least first and second communication paths over a single communication medium, the method comprising:

sampling the signal over the communication paths to realize a first sampling from the first communication path and a second sampling from the second communication path;

synchronizing the first sampling and the second sampling; and

combining the first sampling and the second sampling to generate a signal representative of the signal transmitted through the first and second communication paths.

91. The method as in claim 90 wherein synchronizing the first sampling and the second sampling includes:

independently synchronizing the first sampling and the second sampling; and

adjusting for a delay difference between the first and the second communication paths based on the independent synchronization of the first sampling and the second sampling.

92. The method as in claim 90 wherein the first communication path includes line-neutral path.

93. The method as in claim 90 wherein the second communication path includes a neutral-ground path.

94. A system for receiving a signal transmitted through at least first and second communication paths over a single communication medium, comprising:

means for sampling the signal over the communication paths to realize a first sampling from the first communication path and a second sampling from the second communication path;

means for synchronizing the first sampling and the second sampling; and

means for combining the first sampling and the second sampling to generate a signal representative of the signal transmitted through the first and second communication paths.

95. The system of claim 94 wherein the means for synchronizing the first sampling and the second sampling includes:

means for independently synchronizing the first sampling and the second sampling; and

means for adjusting for a delay difference between the first and the second communication paths based on the independent synchronization of the first sampling and the second sampling.

96. The system of claim 94 wherein the first communication path includes line-neutral path.

97. The system of claim 94 wherein the second communication path includes a neutral-ground path.

98. A computer program stored on a computer readable medium or a propagated signal for receiving a signal transmitted through at least first and second communication paths over a single communication medium, comprising:

a sampling code segment that causes the computer to sample the signal over the communication paths to realize a first sampling from the first communication path and a second sampling from the second communication path;

a synchronizing code segment that causes the computer to synchronize the first sampling and the second sampling; and

a combining code segment that causes the computer to combine the first sampling and the second sampling to generate a signal representative of the signal transmitted through the first and second communication paths.

99. The computer program of claim 98 wherein the synchronizing code segment causes the computer to:

independently synchronize the first sampling and the second sampling; and

adjust for a delay difference between the first and the second communication paths based on the independent synchronization of the first sampling and the second sampling.

100. The computer program of claim 98 wherein the first communication path includes line-neutral path.

101. The computer program of claim 98 wherein the second communication path includes a neutral-ground path.

102. A method of pre-filtering a received signal to improve synchronization, the method comprising:

sampling a randomly located portion of the received signal;

generating a pre-filter based on the randomly located portion of the sampled received signal;

applying the pre-filter to the received signal; and

identifying a data frame preamble based on the received signal after applying the pre-filter to improve synchronization.

103. The method as in claim 102 wherein:

sampling the randomly located portion of the received signal includes sampling within noise; and

generating the pre-filter includes generating the pre-filter based on the sampled noise.

104. The method as in claim 103 wherein the received signal includes data and noise.

105. The method as in claim 102 wherein sampling the randomly located portion of the received signal includes sampling a portion of the received signal at a randomly selected location.

106. The method as in claim 102 wherein the pre-filter includes a hybrid prediction filter that produces a desired response as an inverse of a value related to a portion of the received signal.

107. A system of pre-filtering a received signal to improve synchronization, comprising:

means for sampling a randomly located portion of the received signal;

means for generating a pre-filter based on the randomly located portion of the sampled received signal;

means for applying the pre-filter to the received signal; and

means for identifying a data frame preamble based on the received signal after applying the pre-filter to improve synchronization.

108. The system of claim 107 wherein:

the means for sampling the randomly located portion of the received signal includes means for sampling within noise; and

the means for generating the pre-filter includes means for generating the pre-filter based on the sampled noise.

109. The system of claim 108 wherein the received signal includes data and noise.

110. The system of claim 107 wherein the means for sampling the randomly located portion of the received signal includes means for sampling a portion of the received signal at a randomly selected location.

111. The system of claim 107 wherein the pre-filter includes a hybrid prediction filter that produces a desired response as an inverse of a value related to a portion of the received signal.

112. A computer program stored on a computer readable medium or a propagated signal for pre-filtering a received signal to improve synchronization, comprising:

a sampling code segment that causes the computer to sample a randomly located portion of the received signal;

a generating code segment that causes the computer to generate a pre-filter based on the randomly located portion of the sampled received signal;

an applying code segment that causes the computer to apply the pre-filter to the received signal; and

an identifying code segment that causes the computer to identify a data frame preamble based on the received signal after applying the pre-filter to improve synchronization.

113. The computer program of claim 112 wherein:

the sampling code segment causes the computer to sample within noise; and

the generating code segment causes the computer to generate the pre-filter based on the sampled noise.

114. The computer program of claim 113 wherein the received signal includes data and noise.

115. The computer program of claim 112 wherein the sampling code segment causes the computer to sample a portion of the received signal at a randomly selected location.

116. The computer program of claim 112 wherein the pre-filter includes a hybrid prediction filter that produces a desired response as an inverse of a value related to a portion of the received signal.

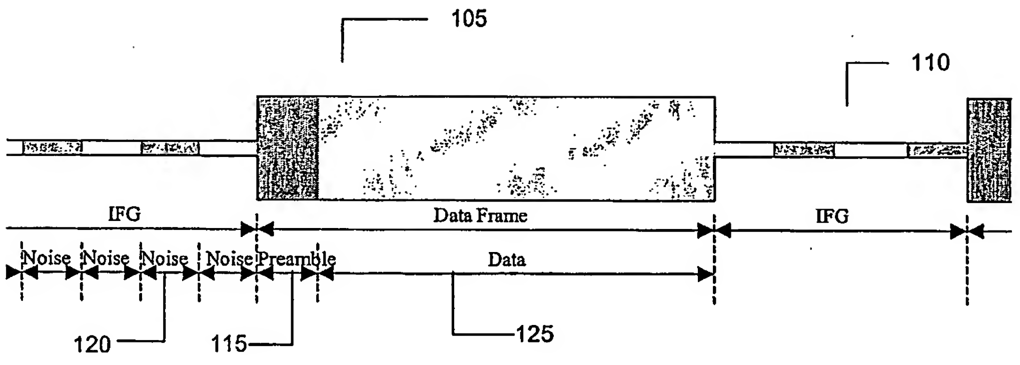

105
110
IFG
Data Frame
IFG
Noise
Noise
Noise
Noise
Preamble
Data
120
115
125

Fig. 1

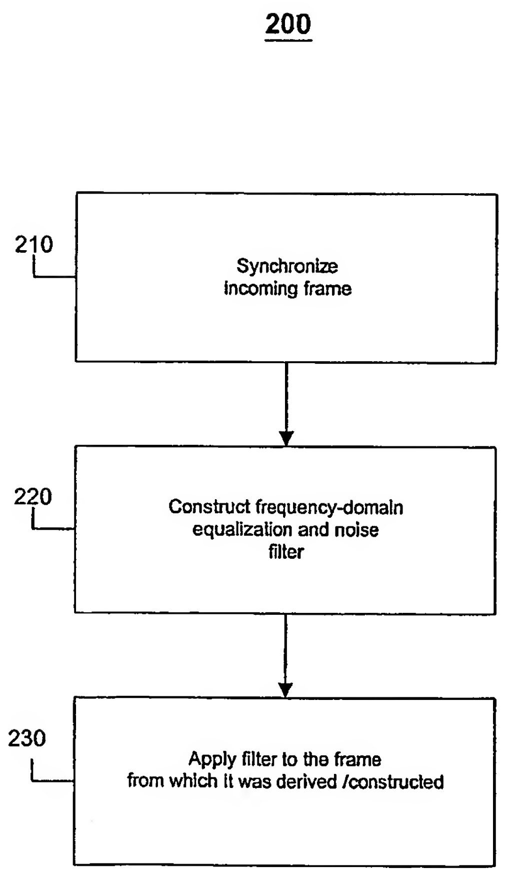

200
210
Synchronize incoming frame
220
Construct frequency-domain equalization and noise filter
230
Apply filter to the frame from which it was derived /constructed

Fig. 2

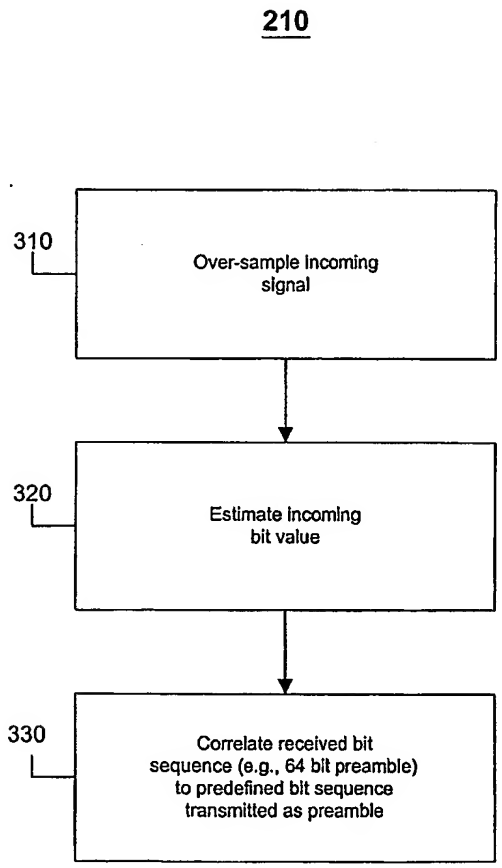

210
310
Over-sample incoming signal
320
Estimate incoming bit value
330
Correlate received bit sequence (e.g., 64 bit preamble) to predefined bit sequence transmitted as preamble

Fig. 3

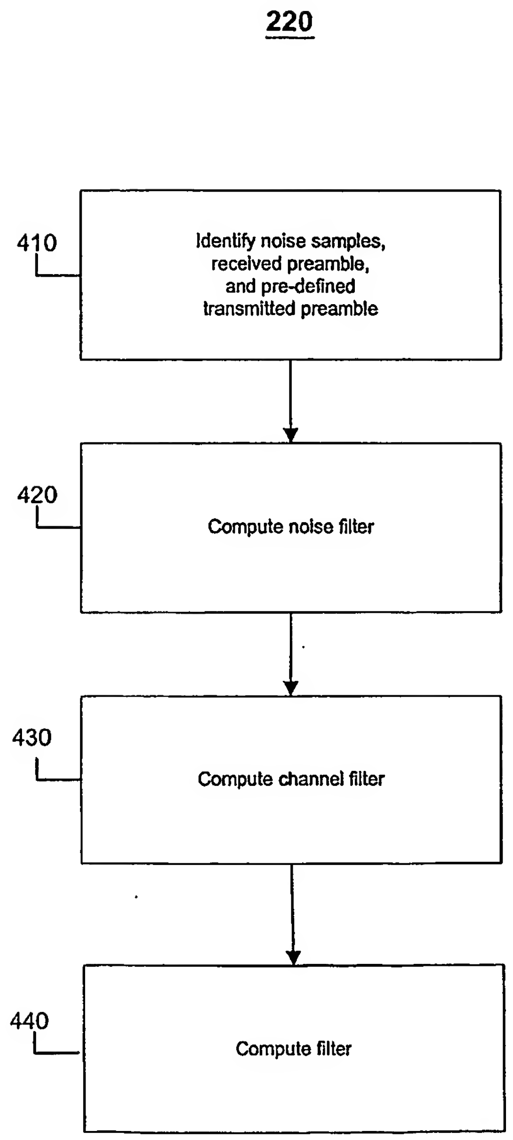

220
410
Identify noise samples, received preamble, and pre-defined transmitted preamble
420
Compute noise filter
430
Compute channel filter
440
Compute filter

Fig. 4

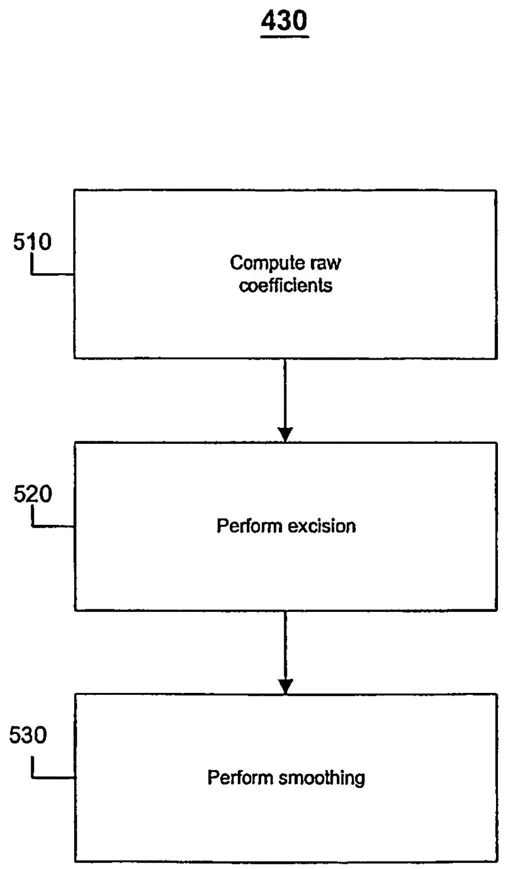

430
510
Compute raw coefficients
520
Perform excision
530
Perform smoothing

Fig. 5

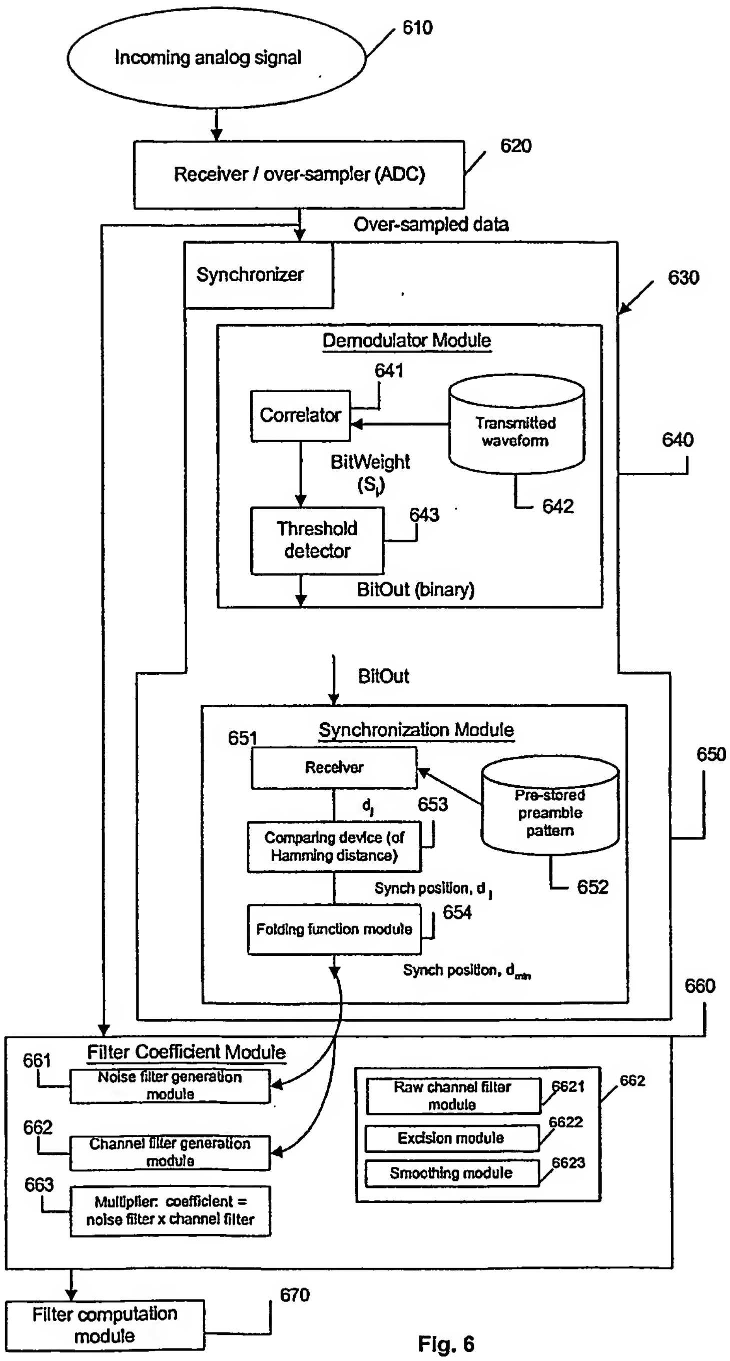
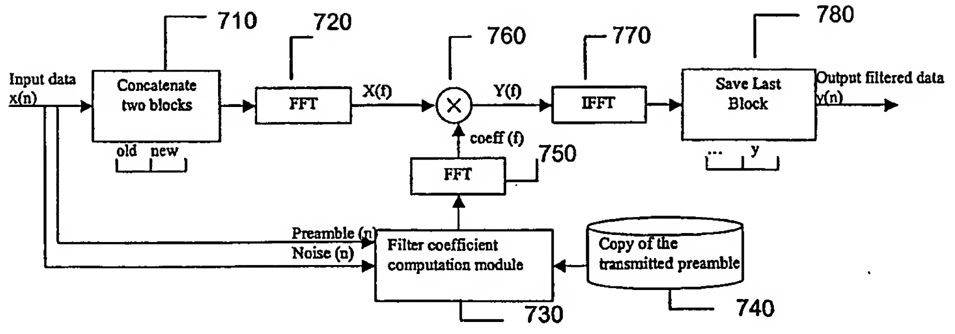

Incoming analog signal
610
Receiver / over-sampler (ADC)
620
Over-sampled data
Synchronizer
630
Demodulator Module
641
Correlator
Transmitted waveform
640
BitWeight (S_i)
642
643
Threshold detector
BitOut (binary)
BitOut
Synchronization Module
651
Receiver
650
Pre-stored preamble pattern
d_j
653
Comparing device (of Hamming distance)
Synch position, d_j
652
654
Folding function module
Synch position, d_{min}
660
Filter Coefficient Module
661
Noise filter generation module
662
Channel filter generation module
663
Multiplier: coefficient = noise filter x channel filter
Raw channel filter module
6621
Excision module
6622
Smoothing module
6623
670
Filter computation module

Fig. 6

700



710
720
760
770
780
Input data x(n)
Concatenate two blocks
old new
FFT
X(f)
Y(f)
IFFT
Save Last Block
... y
Output filtered data y(n)
coeff (f)
FFT
750
Preamble (n)
Noise (n)
Filter coefficient computation module
Copy of the transmitted preamble
730
740

Fig. 7

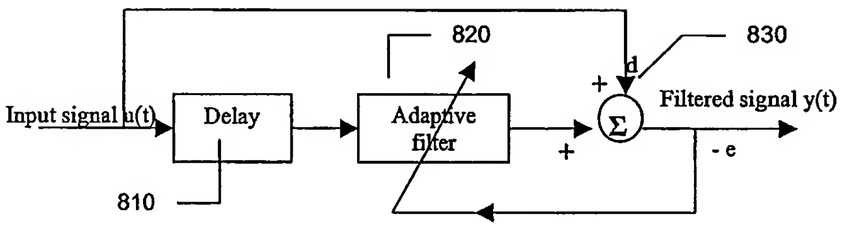

Input signal u(t)
Delay
810
820
Adaptive filter
830
d
+
+
Σ
Filtered signal y(t)
- e

Fig. 8

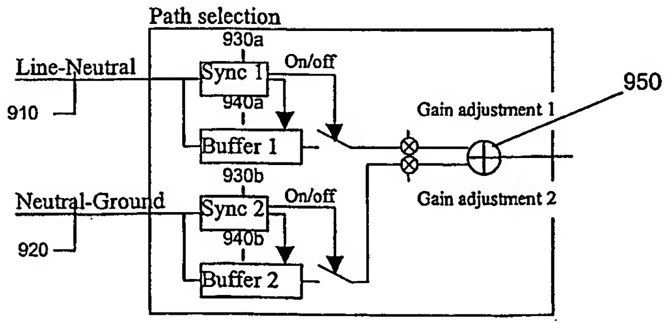

Path selection
Line-Neutral
910
930a
Sync 1
On/off
940a
Buffer 1
Gain adjustment 1
950
930b
Neutral-Ground
920
Sync 2
On/off
940b
Buffer 2
Gain adjustment 2

Fig. 9